НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ
ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмна інженерія»*

на тему: Система обліку сировини та персоналу на металообробного підприємства

Виконав: студент 4 курсу, групи ІТ-51
Бойко Богдан Вікторович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доцент с.н.с, к.т.н. Савчук О. В.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доцент с.н.с, к.т.н. Савчук О. В* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) Інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 *«Програмна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)
“ ”

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Бойку Богдану Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема проекту *«Система обліку сировини та персоналу металообробного підприємства»*

керівник проекту Савчук О. В *с.н.с,к.т.н.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від « » №

2. Термін подання студентом проекту « »

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Розгортання та впровадження програмного забезпечення
4) Керівництво користувача, методика та програма тестування

5. Перелік графічного матеріалу

1) Схема структурна класів програмного забезпечення

2) Структурна схема сайту

6. Консультанти розділів проекту

	Прізвище, ініціали та посада	Підпис, дата	

7. Дата видачі завдання « 13 » березня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання дипломного	Термін виконання	Примітка
1	*Вивчення предметної області*	27.02.2019	
	Аналіз існуючих методів розв'язання		
3	*Постановка та формалізація задачі*	18.03.2019	
	Аналіз вимог до програмного		
	Моделювання програмного		
	Обгрунтування використовуваних		
	Розробка архітектури програмного		
8	*Розробка програмного забезпечення*	05.05.2019	
9	*Налагодження програми*	10.05.2019	
10	*Виконання графічних документів*	16.05.2019	
11	*Оформлення пояснювальної записки*	20.05.2019	
12	*Подання ДП на попередній захист*	04.06.2019	
13	*Подання ДП рецензенту*	13.06.2019	
14	*Подання ДП на основний захист*	18.06.2019	

Студент _____ Бойко Б.В.
(підпис)

Керівник проекту _____ Савчук О.В.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	ІТ51.030БАК.005 ПЗ	Завдання на дипломний проект	1	
2	А4	ІТ51.030БАК.005 ПЗ	Пояснювальна записка	55	
3	А3	ІТ51.030БАК.005 СС	Схема структурна функціональних вимог	1	
4	А3	ІТ51.030БАК.005 СС	Схема структурна бізнес-процесів застосунку	1	
5	А3	ІТ51.030БАК.005 СС	Схема структурна класів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ІТ51.030БАК.005		
Розробн.		Бойко Б.В.			Відомість	Літ.	Лист 1 / Листів 1
Керівн.							
Консульт							

Пояснювальна записка
до дипломного проекту

на тему: Система обліку сировини та персоналу металообробного підприємства

Київ – 2019 року

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з 4 розділів, містить 8 рисунки, 16 таблиць, 12 посилань – загалом 58 сторінки.

Об’єкт дослідження: веб-застосування, що можуть використовуватися для роботи з сировиною та персоналом на металообробному підприємстві.

Мета дипломного проекту: полегшення роботи керівнику та працівнику на підприємстві.

У першому розділі виконано аналіз предметної галузі, відомих технічних рішень, сформульовано функціональні та нефункціональні вимоги до розробленого програмного забезпечення.

У другому розділі описано бізнес-процеси застосунку, моделювання та архітектуру. Описано використання стороннього API.

У третьому розділі описано специфіку тестування та налагодження програмного забезпечення.

У четвертому розділі описано впровадження програмного забезпечення.

У додатках наведено: технічне завдання, керівництво користувача, програма та методика тестування, схема структурна функціональних вимог, схема структурна бізнес-процесів, схема структурна класів програмного забезпечення.

ABSTRACT

The explanatory note of the diploma project consists of 4 sections, 3 pictures, 20 tables, 12 sources – total 53 pages.

The object of study: web applications working statistic.

The aim of the diploma project: increase the effectiveness of customer and employement.

In the first section, the subject area was analyzed, existing technical solutions were investigated, functional and non-functional requirements for the developed software were formulated.

In the second section, the business processes were defined. Application modeling, architecture and using outside API were described.

The third section describes the specification of application testing.

The fourth section describes the application implementation.

Annexes contain the terms of reference, the user's guide, testing program and methodic, the structural diagram of functional requirements, the diagram of business processes, the structural class diagram.

Keywords: ANALYTICS, TEXT RECOGNITION, REACT, APPLICATION, USER SEARCH, MATERIAL SEARCH

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

API – набір визначень для взаємодії різнотипного програмного забезпечення.

JSX – розширення синтаксису JavaScript, яке виглядає як XML.

EXEL – формат табличного документу

JPG, JPEG, PNG – формати зображень.

XML – розширювана мова розмітки.

NOSQL – база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційних базах даних

ВСТУП

У наш час все більше підприємств переходять на автоматизовані системи управління, що значно полегшує збереження, обробку та аналіз даних про стан на підприємстві. На основі цих даних керівник чи бухгалтер можуть швидко виявити та видалити недоліки в процесах на підприємстві, щоб досягти максимальної продуктивності. Автоматизована система управління зберігає всю інформацію на одному з серверів підприємства, що в свою чергу допомагає полегшити документообіг на підприємстві.

Керівник підприємства має актуальну інформацію про стан свого підприємства, а бухгалтер отримує більшість звітів про підприємство за будь-який час роботи системи на підприємстві. Зникає необхідність в збереженні інформації в паперовому вигляді та продовження роботи архівів.

У сучасному світі уже створено безліч систем для автоматизації окремих відділів підприємства, але не було запропоновано об’єднати ці системи в одну. У такій системі можна вести облік із різних сфер підприємств, наприклад: бухгалтерська справа, відділ кадрів та інші. Завдяки такому підходу є можливість цілісно оцінювати наше підприємство.

Мета цієї роботи – створення автоматизованої системи обліку сировини та персоналу металообробного підприємства, що допоможе в управлінні підприємством та автоматизації процесів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Недоліки системи обліку, що існує, та ідея її інтелектуалізації

Із розвитком науково-технічного прогресу в галузі інформатики та менеджменту виникла ідея підвищення рівня інтелектуалізації обліку сировини та персоналу, яка була не пристосована до потреб керівництва та менеджменту в сучасному світі.

Інтелектуалізація системи обліку виносить на перший план пізнання навколишнього світу за допомогою інтелекту, що здобув певні знання за допомогою попередніх запитів до системи. Джерелом знань для інтелекту може також бути інформація про навколишній світ та середовище, у якому він знаходиться.

В індустріальному суспільстві науковці створювали універсальні моделі обліку на основі ручної праці та механізації найбільш трудомістких процесів з обробки та аналізу даних, то в постіндустріальному суспільстві на перший план виносять індивідуальні потреби користувача та сучасні методи збору та обробки інформації.

Досить актуальною проблемою будуть залишатися генерування можливого більшого обсягу інформації з метою вибору варіантів управління. Облік посилює свою сервісну функцію для системи менеджменту, що й означає посилення його інтелектуальності та науковості.[1]

1.1.1 Аналіз відомих технічних рішень

Існує декілька відомих технічних рішень для створення клієнтської частини системи. Далі розглянемо 3 основні з них.

– React. Відома бібліотека для створення інтерфейсів користувача, яка здатна вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і більшістю індивідуальних розробників. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані,

котрі змінюються з часом без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з Redux. React обробляє тільки інтерфейс користувача в застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC, MVVM та MVX), і може бути використане у поєднанні з іншими JavaScript бібліотеками або фреймворках MVC.[2]

– AngularJS. Фреймворк з відкритим програмним кодом, який розробляє Google. Призначений для розробки односторінкових додатків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета — розширення браузерних застосунків на основі шаблону Модель-вид-контролер (MVC), а також спрощення їх тестування та розробки. Фреймворк працює зі сторінкою HTML, що містить додаткові атрибути і пов'язує області вводу або виводу сторінки з моделлю, яка є звичайними змінними JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON-даних.[3]

– Vue.js. Фреймворк створений Коли Еван Ю, який працював в Google Creative Labs. Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Усередині Vue компілює шаблони в рендерингові функції віртуального DOM. У поєднанні з реактивною системою Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться. У Vue ви можете використовувати синтаксис шаблонів або напряму писати рендерингові функції використовуючи JSX. Одна із найвиразніших особливостей Vue — це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти.[4]

Порівняльна характеристика технічних рішень наведена в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика наявних рішень

Назва	Сервер рендеринг	Підтримка Typescript	Можливість повного безкоштовного використання
React	Так	+	+
AngularJS	Ні	+	+
Vue.js	Ні	-	+

1.1.2 Аналіз відомих програмних продуктів

Існує досить багато систем для автоматизації процесів на підприємстві. Далі розглянуті найвідоміші з них.

– 1с Підприємство. Головні модулі є технологічним блоком, що реалізує загальну функціональність, яка залежить від технічних особливостей програмного та апаратного забезпечення. Ці модулі здійснюють також підтримку так званих "конфігурацій" і є базою для виконання програм на вбудованій мові програмування. Конфігурація – це множина ключових змінних, тобто тих, що статично характеризують підприємство та його діяльність (код реєстрації, ставка ПДВ, план рахунків, нормативні показники тощо), та "бізнес-правил" у вигляді програм на вбудованій мові високого рівня, які характеризують бізнес-процеси підприємства в динаміці. Перевагою АІС є здатність до гнучкого конфігурування. До недоліків системи можна віднести високу вартість впровадження та експлуатації, а також надто складну будову конфігурацій, для освоєння якої потребується тривале навчання спеціалістів. Для комплексної автоматизації підприємства і виконання масштабних і складних проектів рекомендується запрошувати партнерів зі статусами 1С:Центр компетенції з корпоративних рішень (1С:КОРП) і 1С:Центр компетенції по ERP-рішень (1С:ERP). У деяких конфігураціях майже відсутній захист інформації від несанкціонованого доступу, що не дозволяє використовувати систему на підприємства де потрібно зберігати досить секретні матеріали.[5]

– SAP ERP. Популярна система управління комплексного контролю діяльності та інформаційних потоків підприємства, що створена німецькою компанією SAP AG. Найбільш відомий продукт - модулі ERP. Це – програмний інструментарій, який функціонує на базі одиничних баз даних, створюючи загальне інформаційне поле всередині компанії. Основна задача ERP-систем - забезпечення безперервної, взаємозалежної, комплексної автоматизації всіх функціональних учасників, підрозділів і бізнес-процесів підприємства. Їх, як правило, використовують у масштабних корпораціях, на великих виробничих комплексах.[6]

– Microsoft Dynamics. Лінійка продуктів програмного забезпечення для бізнесу від корпорації Microsoft. Спочатку вона існувала під кодовою назвою Project Green. А у вересні 2005 року Microsoft Dynamics замінила старий бренд Microsoft Business Solutions. В Україні Microsoft Dynamics представлена такими рішеннями: Microsoft Dynamics AX, Microsoft Dynamics NAV та Microsoft Dynamics CRM. Напрямок Microsoft Business Solutions тут було офіційно відкрито у травні 2006 року. Наразі рішення на базі Microsoft Dynamics використовуються у понад 100 українських компаніях. [7]

Усі програмні продукти, описані вище, вимагають створення окремої системи для різних підрозділів, що змушує одночасно мати декілька систем на одному підприємстві. Також системи не мають в собі програми для відслідковування часу роботи працівника. Тому створення веб-застосунку який працював би відразу з усіма підрозділами на підприємстві є актуальним.

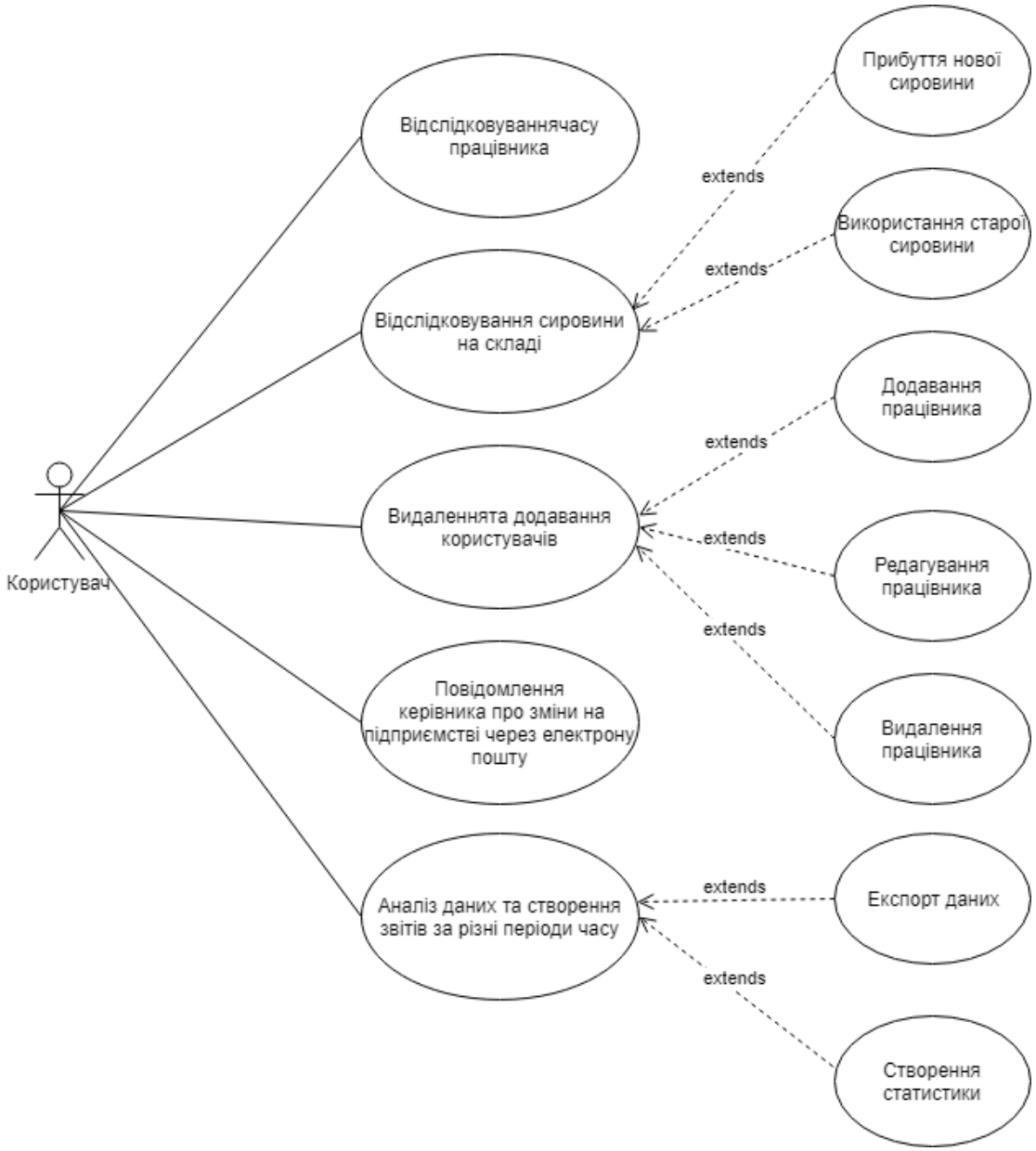
1.2 Аналіз вимог до програмного забезпечення

1.2.1 Розроблення функціональних вимог

Веб-застосунок має безпосередньо одного користувача. У ході аналізу були виявлені такі необхідні користувачу варіанти використання:

– відслідковування часу працівника;
– відслідковування сировини на складі;
– видалення та додавання працівників;

– повідомлення керівника про зміни на електронну пошту;
– аналіз даних та створення звітів за різні періоди часу.



Варіанти використання зображені на рисунку 1.1. вимоги зазначені у таблиці 1.2.

Рисунок 1.1 – Схема структурних варіантів використання

Таблиця 1.2 – Функціональні вимоги

Варіант Використання	Функціональна вимога	Приорітет

Продовження таблиці 1.2

Відслідковування відпрацьованого часу Працівника	1. Застосунок має надавати можливість починати та зупиняти відлік часу працівника 1.1 Застосунок має надавати можливість переглядати	Високий
Відслідковування сировини на складі	2. Застосунок має надавати можливість змінювати кількість сировини на складі 2.1 Застосунок має надавати можливість редагування кількість сировини на складі 2.2 Застосунок має надавати можливість додавати нову сировину 2.3 Застосунок має надавати можливість видаляти використану сировину	Високий
Видалення та додавання Працівників	3. Застосунок має надавати можливість видаляти звільнених працівників та додавати нових 3.1 Застосунок надавати можливість редагувати профіль користувача	Високий
Повідомлення керівника про зміни на підприємстві через електрону пошту	4. Застосунок має надсилати листа при будь-яких змінах на підприємстві	Високий

Продовження таблиці 1.2

Аналіз даних та створення звітів за різні періоди часу	5. Застосунок має надавати можливість створювати та переглядати звіти за різні проміжки часу 5.1 Застосунок має відображати статистику сировини та працівників 5.2 Застосунок має надати можливість завантажити звіт за різні періоди роботи підприємства	Високий

1.2.2 Розроблення не функціональних вимог

Для розробки веб-застосунку було обрано React для клієнтської частини та Node.js для серверної. За даними сервісу Google Trends, за останні 2 роки React є однією з найбільш популярних бібліотек.[8] Для розробки інтерфейсу для користувача було обрано Materail UI від Google, що набирає популярність в останні роки. Розробка веб-застосунку буде виконана на мові JavaScript з використанням бази даних MongoDB, яка легко інтегрується з Node.js. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД). Веб-застосунок буде створено з урахуванням сучасних вимог користувачів та використанням кращих практик в розробці. Проаналізувавши основні вимоги, було виділено наступні нефункціональні вимоги:

- веб-застосунок має працювати в усіх сучасних браузерах;
- веб-застосунок має коректно відображатися на всіх екранах ПК;
- передбачено захист від некоректних дій користувача;
- повинен мати інтуїтивно зрозумілий інтерфейс;

– дані про користувача та сировину збережено від зломистників;
– система попереджує про збереження чи видалення інформації.

1.3 Постановка комплексу завдань розробки

Метою розробки є полегшення роботи керівника з великим об’ємом інформації. У системі обліку буде збережено інформацію про працівників та сировину компанії, що значно полегшить роботу з виплатою заробітної плати та перевірку кількісті використаної сировини за різні періоди часу. Працівник підприємства в свою чергу зможе легко відслідковувати відпрацьований час та кількість часу витраченого на кожну виконану задачу.

Для досягнення поставленої мети повинні бути вирішені наступні задачі:

– створення модуля для онлайн збереження та редагування інформації про сировину на складі та кількість працівників на підприємстві. Модуль має надати статистику надходження та використання сировини

– створення модулю роботи з базою даних. Модуль має надавати можливості збереження, вибірки та оновлення даних про сировину;

– створення зручного інтерфейсу користувача для задоволення функціональних вимог. Інтерфейс має бути зручним і легким у використанні.

1.4 Висновки до розділу

У ході проведення аналізу вимог до розробки було детально розглянуто та проаналізовано предметне середовище – системи обліку на підприємсті та сучасні технології для розробки веб-застосунків, оглянуті наявні рішення (React, Angular, Vue.js) та популярні програмні продукти (1с Підприємство, SAP ERP, Microsoft Dynamics). Виявлено, що усі програмні продукти, описані вище, вимагають створення окремої системи обліку для кожного з підрозділів і не мають можливості відслідковувати кількість відпрацьованих годин працівником та час виконання окремих завдань. Тому було вирішено, що необхідно створити таку систему обліку персоналу та сировини, яка задовільнила би як керівництво підприємства, так і персонал. Далі було досліджено сценарій користувача та керівника, розроблені функціональні та нефункціональні вимоги.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Нижче описані загальні принципи, за якими має працювати застосунок:

а) перше використання;

1) створити профіль керівника підприємства та вивести форму заповнення профіля;

2) вивести базову статистику підприємства;

б) кожне наступне використання;

1) сформувати нову статистику, згідно змін що відбулись за відсутності керівника;

2) сформувати новий звіт сировини на складі та працівників на підприємстві

в) натиснення на «Users» в боковому меню;

1) вибрати з бази даних усіх користувачів, відсортувавши за статусом активності;

г) натиснення на «Export» у верхньому кутку таблиці на сторінці працівників;

1) запропонувати формат файла xls/csv;

2) завантажити файл;

д) натиснення на «Materials» в боковому меню;

1) вибрати з бази даних усіх користувачів, відсортувавши, за статусом активності;

е) натиснення на «Export» у верхньому кутку таблиці на сторінці сировини;

1) запропонувати формат файла xls/csv;

2) завантажити файл;

ж) натиснення на «Dashboard» в боковому меню;

1) проаналізувати та відобразити статистику працівників;

2) проаналізувати та відобразити статистику сировини на складі

підприємства.

Основні бізнес-процеси застосунку відповідають основним принципам роботи веб-застосунку, наведеним у попередньому підрозділі, та включають в себе створення нових працівників та редагування їх профілю, додавання та видалення сировини з обліку підприємства. Створено сторінку для підрахунку відпрацьованих годин працівником.

Схеми структурні бізнес-процесів, представлені у вигляді BPMN нотації, наведені у Документі ІТ51.030БАК.005 СС Схема структурна бізнес-процесів застосунку.

2.2 Архітектура програмного забезпечення

Застосунок має містити 3 окремі модулі, які відповідають за відповідні задачі:

- модуль збору та аналізу даних;
- модуль роботи з базою даних;
- модуль інтерфейсу користувача.

2.2.1 Аналіз середовища виконання програмного забезпечення

Для розробки веб-застосунку під платформу всі платформи потрібно мати встановлену одну з сучасних операційних систем MacOS, Windows або Linux. В системі має бути встановлений один з сучасних браузерів Chrome, Firefox або Opera.

В основі системи обліку персоналу та сировини металообробного підприємства буде використано сучасну версію React 16.3. У цій версії було змінено основні методи відображення та оновлення наших компонентів. Компоненти аналогічно функціям JavaScript зберігають інформацію про свій стан та взаємодіють з зовнішніми запитами, їх життєвий цикл зображено на рисунку 2.1.

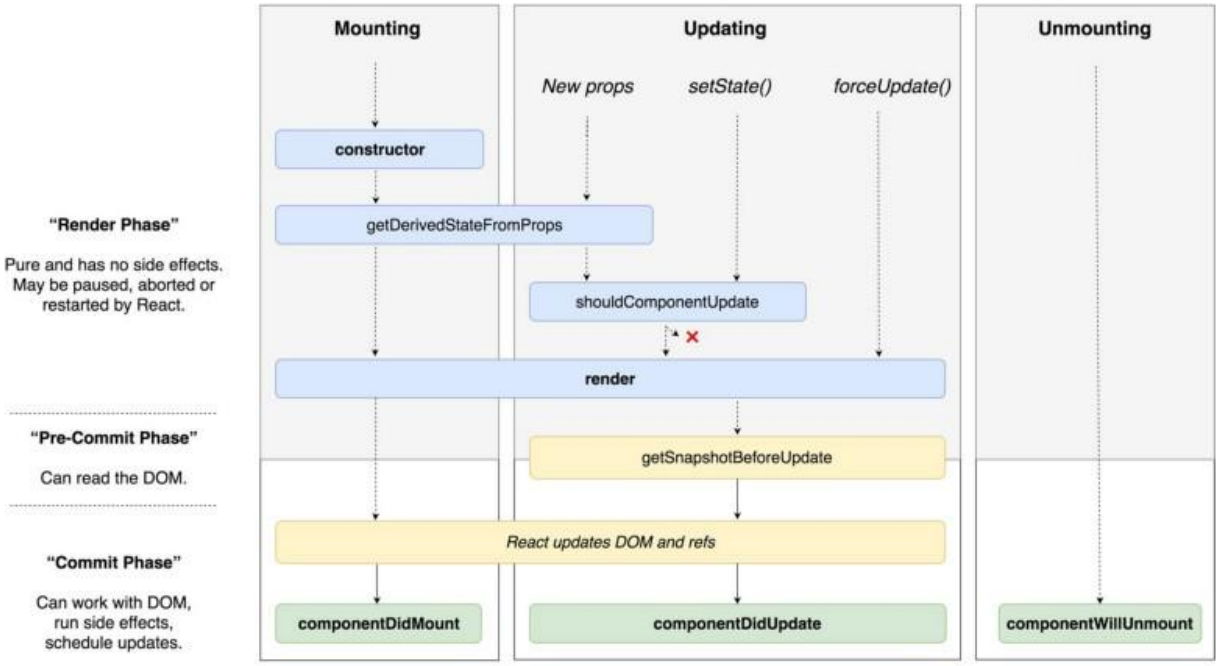


Рисунок 2.1 – Спрощене зображення життєвого циклу React компонента

Для зручності систему обліку персоналу та сировини буде розбито на маленькі частини, в React це Component. Component – це невід'ємна частина при побудові будь якого застосунку, він зберігає всю інформацію та всі можливі взаємодії з ними. Також в компонент буде поміщено всі його залежності та CSS.

React заснований на віртуальному DOM - у пам'яті представлено те, як DOM виглядає насправді. Дані в React значною мірою є незмінними, і DOM-зміни розраховуються на основі визначення відмінностей. Підхід з віртуальним DOM забезпечує функціональний спосіб описати уявлення в будь-який момент. Тому що не використовується шаблон спостерігача і не перемальовується увесь додаток при кожному оновленні, уявлення щодо визначення гарантує бути синхронним з даними, що також відкриває можливості для ізоморфних JavaScript-додатків. Знаючи те, що React виконує всі свої дії спочатку в Virtual DOM і тільки потім переносить зміни в DOM, потрібно обережно користуватися наступними методами componentWillMount та componentWillUnmount. В даний момент можемо скористатися можливостями для реєстрації ref, а також додати це місце, де ми хотіли б вказати установку фокусу. Так, таймату, ajax-запит і взаємодія з іншими бібліотеками потрібно оброблювати в цьому методі.

Застосунок складається з 3 модулів, що відповідають за окремі функціональні частини роботи – User Interface Module, Adress Module, API Module. Опис спроектованих модулів та класів наведено у таблиці 2.1.

Застосунок складається з 3 модулів, що відповідають за окремі функціональні частини роботи – User Interface Module, Recognition Module, API Module. Опис спроектованих модулів та класів наведено у таблиці 2.1.

Застосунок складається з 3 модулів, що відповідають за окремі функціональні частини роботи – User Interface Module, Recognition Module, API Module. Опис спроектованих модулів та класів наведено у таблиці 2.1.аываыв

2.2.2 Опис архітектури програмного забезпечення

Застосунок складається з 3 модулів, що відповідають за окремі функціональні частини роботи – User Interface Module, Adress Module, API Module. Опис спроектованих модулів та класів наведено у таблиці 2.1.

Таблиця 2.1 – Опис основних класів додатку

Модуль	Клас	Призначення
Adress module	Adress Helper	Відповідає за роботу з Google Address API
API module	DatabaseHandler	Відповідає за роботу з запитами до серверу
	SortFilter	Відповідає за формування запиту заданим фільтрам та параметрам
User Interface Module	App	APP відповідає за інтерфейс користувача

Продовження таблиці 2.1

	LoginFragment	Фрагмент, що відображається при першому вході у застосунок
	StatisticFragment	Фрагмент, що відображається коли виконується певний аналіз даних або пошук
	UserFragment	Фрагмент, що відображає користувача у системі та повідомляє про нових користувачів, яких необхідно активувати
	SearchFragment	Фрагмент для пошуку працівника за текстом та статусом
	SearchResultFragment	Фрагмент, що відображає результат пошуку,
	ProfileFragment	Фрагмент, що відображає зображення та інформацію працівника, його завдання та кількість відпрацьованих годин за кожен відпрацьованого дня

Продовження таблиці 2.1

	AddNewUser Fragment	Фрагмент, що дозволяє створювати нових працівників та відсилати лист на його електрону пошту

2.2.3 Архітектурні компоненти програмного забезпечення

Модуль роботи з базою даних. Бібліотека React не має вбудований інструментарій для управління базою даних sqlite3, тому будемо взаємодіяти з базою через наш сервер. Модуль веб-застосунку, що відповідає за роботу з базою даних на сервері, складається з 3 класів – DatabaseHandler, Exporter та UserTime.

DatabaseHandler – клас, що відповідає за зв’язок з базою даних. Він містить методи, що дозволяють створити таблицю для збереження записів, вибрати записи з бази даних, додати записи, редагувати або видалити. Пошук інформації за текстом у базі даних відбувається через співпадіння тексту.

Exporter – клас, що перевіряє актуальну інформацію про наявність матеріалів та працівників у базі даних. На основі отриманої інформації створює Exel файл з всіма полями що є в базі.

UserTime – клас, що відображає кількість відпрацьованих годин працівником. Його продуктивність на завданнях що було видано керівником. Його опис наведений у таблиці 2.2

Таблиця 2.2 – Опис сутності UserTime

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор

Продовження таблиці 2.2

userLocation	String	-	Місце де працівник проживає
name	String	-	Ім’я користувача в системі
hoursWork	int (0 або 10)	-	Кількість фактично відпрацьованих годин на працівником

Опис основних методів класів модулю Database наведений у таблиці 2.3.

Таблиця 2.3 – Опис основних методів класів модулю Database

Класс	Метод Параметри виклику Тип значення, що повертається	Опис дій
DatabaseHandler	addUserInfo userInfo User void	Метод додає інформацію про працівника до бази даних
DatabaseHandler	getUser int id UserInfo	Метод повертає інформацію про працівника за заданим id
DatabaseHandler	getUserLocation int id UserInfo	Метод повертає інформацію про місцезнаходення працівника за заданим id

Продовження таблиці 2.3

DatabaseHandler	deleteByID int id void	Метод видаляє інформацію про працівник за заданим ідентифікатор
DatabaseHandler	getAllUsers - ArrayList<UserInfo>	Метод повертає інформацію про всіх працівників в таблиці UserInfo
DatabaseHandler	updateUser UserInfo user int	Метод оновлює інформацію про працівника
DatabaseHandler	addMaterials int id void	Метод додає інформацію про про матеріал.
DatabaseHandler	getMaterial int id MaterialInfo	Метод повертає інформацію про працівника за заданим id
DatabaseHandler	getAllMaterials - Array<MaterialInfo>	Метод повертає кількість про всіх матеріали в таблиці UserInfo
DatabaseHandler	getStatisticUser - ArrayList<StatUser>	Метод повертає інформацію про статистику працівників
DatabaseHandler	getStatisticUser - ArrayList<StatUser>	Метод повертає інформацію про статистику працівників

Продовження таблиці 2.3

DatabaseHandler	getStatisticMaterials - ArrayList<StatMaterial>	Метод повертає інформацію про статистику матерілів на складі
DatabaseHandler	getTeam - ArrayList<Team>	Метод оновлює інформацію про розділи та підрозділи працівників
DatabaseHandler	SortUsersBy String sortParam ArrayList<UserInfo >	Метод повертає список працівників відсортованих за переданим кретерієм
DatabaseHandler	FilterUser String filterParam ArrayList<UserInfo >	Метод повертає список працівників що мають текст за параметрами
DatabaseHandler	SortMaterialsBy String sortParam ArrayList< MaterialInfo >	Метод повертає список матеріалів відсортованих за переданим кретерієм
DatabaseHandler	FilterMaterials String sortParam ArrayList< MaterialInfo >	Метод повертає список матеріалів що мають текст за параметрами
Exporter	getUserTableEXEL Date dateRange Exel File	Метод повертає Exel файл за заданою датою
Exporter	getMaterialTableEXEL Date dateRange Exel File	Метод повертає Exel файл за заданою датою

Продовження таблиці 2.3

Exporter	getStatisticEXEL Date dateRange Exel File	Метод повертає Exel файл з статистикою за проміжок часу
UserTime	setStartTime Date start String	Метод починає відлік часу коли працівник зайшов в системи
UserTime	setStopTime Date start String	Метод зупиняє відлік часу коли працівник вийшов з системи
UserTime	getCurrentWorkTime int id Date time	Метод повертає кількість відпрацьованих годин
UserTime	getStatisticTime int id ArrayList<UserTime>	Метод повертає статистику Відпрацьованих годин за день
UserTime	removeTime int id String	Метод видаляє проміжок часу вибраний працівником
UserTime	editTime int id String	Метод змінює час роботи працівника

Для захисту від некоректних дій користувача з роботою з базою даних під час здійснення заповнення форм для запрошення працівників та редагування данних про сировину на складі, було використано сучасні методи, що захищають дані від sql-ін’єкцій.

Для коректної роботи цих методів було створено спеціальний валідатор для бази даних. Опис основних методів класів StatisticHelper наведений у таблиці 2.4.

Таблиця 2.4 – Опис основних методів класів модулю StatisticHelper

Класс	Метод Параметри виклику Тип значення, що повертаєтсья	Опис дій
StatisticHelper	collectUserStatistic - ArrayList<UserStat>	Метод повертає заданий масив статистики працівників
StatisticHelper	collectMaterialStatistic - ArrayList<MaterialStat>	Метод повертає заданий масив використання сировини
StatisticHelper	createUserTimeStatistic Date startDate Date endDate ArrayList<Date>	Метод повертає заданий масив часових проміжків
StatisticHelper	checkWorkDate Date startDate Date endDate ArrayList<Date>	Метод повертає всі робочі години за заданий проміжок часу
StatisticHelper	newMaterialStat - ArrayList< MaterialStat >	Метод повертає всі додані матеріали за останій місяць

Модуль користувацького інтерфейсу. Модуль складається з View та Fragments, що відповідають необхідним частинам поведінки застосунку. Опис основних методів класів наведений у таблиці 2.5

Таблиця 2.5 – Опис основних методів класів модулю UI

Класс	Метод Параметри виклику Тип значення, що повертається	Опис дій
ImageFragment	newInstance String imageLocation ImageFragment	Метод повертає новий фрагмент. Стандартний метод
ImageFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState View	Метод створює та повертає ієрархічне представлення фрагменту. Стандартний метод
Textragment	onCancelClick - void	Метод відміняє редагування тексту
Fragment	onSaveClick - void	Метод зберігає текст інформацію заповненої форми в базу даних
Fragment	onDeleteClick int id void	Метод видаляэ інформацію про заданого працівника

Продовження таблиці 2.5

MainActivity	onCreate savedInstanceState void	У методі виконуються попередні дії для запуску застосунку. Стандартний метод
MainActivity	onBackButtonPressed - void	Метод оброблює натиснення на кнопку “назад”. Стандартний метод
MainActivity	newInstanceProcess ArrayList<String> imagesToProcess, ArrayList<String> imagesToDelete ProcessFragment	Метод створює новий екземпляр класу ProcessFragment
ProcessFragment	newInstanceSearch String search_text ProcessFragment	Метод створює новий екземпляр класу ProcessFragment
ProcessFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle	Метод створює та повертає ієрархію фрагменту

Продовження таблиці 2.5

ProcessFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState View	Метод створює та повертає ієрархічне представлення фрагменту. Стандартний метод
ProcessFragment	onViewCreated View view, Bundle savedInstanceState void	Метод, який викликається після завершення створення відображення. Стандартний метод
ProcessFragment	taskProcess final ArrayList<String> imagesToProcess, final ArrayList<String> imagesToDelete void	Метод відображає процес знаходження тексту на зображеннях
ProcessFragment	taskSearch String mSearchText void	Метод відображає процес пошуку зображення за вмістом тексту

Продовження таблиці 2.5

RealTimeRecognitionFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState View	Метод створює та повертає ієрархічне представлення фрагменту. Стандартний метод
RealTimeRecognitionFragment	onViewCreated @NonNull View view, @Nullable Bundle savedInstanceState void	Метод, який викликається після завершення створення відображення. Стандартний метод
RealTimeRecognitionFragment	onTextClick - boolean	Метод копіює розпізнаний текст до буфера обміну
SearchFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState View	Метод створює та повертає ієрархічне представлення фрагменту.
SearchFragment	search - void	Метод викликає відображення пошуку

Продовження таблиці 2.5

SearchFragment	real_time - void	Метод викликає розпізнавання тексту в реальному часі
SearchFragment	newItems - void	Метод викликає відображення вибраних працівників
SearchFragment	onTextChanges - void	Метод, який буде викликаний при зміні тексту пошуку. Стандартний метод
SearchResultFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState View	Метод створює та повертає ієрархічне представлення фрагменту. Стандартний метод
SearchResultFragment	newInstance ArrayList<String> images SearchResultFragment	Метод створює новий єкземпляр фрагменту результатів пошуку.

Продовження таблиці 2.5

StartFragment	newInstance ArrayList<String> imagesToProcess StartFragment	Метод створює новий єкземпляр фрагменту старту. Стандартний метод
StartFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState View	Метод створює та повертає ієрархічне представлення фрагменту. Стандартний метод
StartFragment	start - void	Метод розпочинає використання застосунку при першому запуску
UpdateFragment	newInstance ArrayList<String> imagesToProcess StartFragment	Метод створює новий екземпляр фрагменту старту. Стандартний метод
UpdateFragment	onCreateView LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState	Метод створює та повертає ієрархічне представлення фрагменту.

Продовження таблиці 2.5

UpdateFragment	skipUpdate - void	Метод, який продовжує використання застосунку без оновлення бази даних
UpdateFragment	update - void	Метод оновлює базу даних перед початком використання застосунку

Основні використані елементи React:

– React Class – клас що відповідає за відображення та поведінку користувацького інтерфейсу;

– Fragment – частина інтерфейсу або поведінки;

– Layout – елемент, що прив’язує макет XML-файлу у відповідний об'єкт View;

– Package – зв’язка ключів параметрів та значень параметрів;

– ViewGroup – спеціальний View, що служить контейнером для інших View;

– Material Table – спеціальний Table, що містить в собі список пагінатор та надає можливість роботи з ними;

– Toolbar – адаптер для роботи з декількома елементами;

– Table Row – адаптер для роботи з даними в рядках;

– Imag Preview– класс для додавання та редагування зображеннь в веб-застосунку;

Основні використані елементі розмітки XML-макету:

– Layout, RelativeLayout – макети розмітки;

– App.Text, App.EditText– елементи макету;

– ScrollBar – елемент для створення «прокручування» змісту екрану.

Застосунок також містить наступні файли ресурсів:

– strings.xml – рядкові ресурси (текст);

– colors.xml – кольорова гама застосунку;

– dimens.xml – розміри та відстані між елементами.

Винесення усіх значень змінних, що відповідають за відображення на екрані, в окремі файли полегшує роботу з ними. Під час розробки застосунку розробник лише посилається на такі змінні, а React сам «підтягує» їх значення та автоматично налаштовує їх роботу. Це полегшує роботу розробку, адже в разі зміни значення якоїсь змінної її треба буде змінити лише один раз у одному місці. Для текстових ресурсів таким чином можна указати декілька варіантів локалізації.

Також основою будь-якого React-застосунку є файли package.json та package-lock.json, котрі містить важливу інформацію про веб-застосунок. Manifest містить перелік всіх Activity застосунку з їх діями, мінімальний рівень пакетів, необхідний для роботи застосунку, дозволи, які потрібні застосунку для роботи (для Profile – це дозвіл на читання зображення у пам’яті пристрою). А package. json – всю необхідну інформацію для збирання проекту, в тому числі залежності використаних біблиотек.

Для повноцінного функціонування застосування було використано бібліотеки з відкритим вихідним кодом:

– Redux – бібліотека для збереження глобальної інформації про стан веб-застосунку, наприклад дані користувача.

– Material UI - бібліотека для зручної роботи з View.

При розробці була використана система автоматичного контролю версій git.

2.3 Висновки до розділу

Застосунок складається з 3 модулів, що відповідають за необхідні

частини роботи застосунку.

Модуль інтерфейсу користувача складається з Activity та Fragments. Він розроблений відповідно до типової архітектури застосунку React та надає користувачу зрозумілий та зручний інтерфейс для роботи з застосунком.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування застосунку є невід'ємною частиною процесу розробки. Тестування веб-застосування - складний процес: різні розміри екрану, апаратні відмінності, кілька версій операційних систем, різні типи підключення до інтернету, раптові обриви зв'язку тощо. Для тестування інженери QA використовують різні інструменти тестування та хмарні служби. Використовувані інструменти визначають методологію тестування.

Послідовно виконуючи тести застосунку, можна підтвердити правильність, функціональну поведінку та зручність використання, перш ніж публікувати його.

Тестування також надає такі переваги:

– швидкий відгук про помилки;

– раннє виявлення несправностей у циклі розробки;

– рефакторинг безпечного коду, що дозволяє оптимізувати код;

– стабільна швидкість розвитку, що допомагає мінімізувати технічні втрати.

Тестування веб-застосунків має свої особливості:

– існує велика кількість браузерів;

– для тестування потрібні різні моделі пристроїв;

– існують різні розміри екранів;

– основна функція ПК – виконання основних сервісів та програм операційної системи, тому застосунок не повинен втручатися в ці важливі функції;

– широкий спектр конкретних операційних систем та компонентних конфігурацій;

– операційна система пристрою швидко старіє;

– виходять нові версії JavaScript;

– ПК використовують кабельне підключення (LAN)

– ПК пристрої постійно здійснюють пошук мережі. Треба тестувати застосунок з різною швидкістю передачі даних;

– веб-застосунки підтримують одразу кілька вхідних каналів (клавіатура, голос, жести і т. д.) ;

Дотримання найкращих практик тестування веб-застосунків та ретельна перевірка всієї функціональності застосунку є дуже важливим для успіху застосунку.

3.2 Опис процесу тестування

Як було вказано вище, тестування веб-застосунків має свої особливості. У тестуванні веб-застосунку існує три категорії тестів: малі, середні та великі. [12]

Малі тести - одиничні тести, які можна виконувати окремо від виробничих систем. Вони, як правило, покривають код кожного основного компоненту і повинні швидко працювати на будь-якому комп'ютері. Таке тестування виконується засобами React Unit Testing.

Середні тести - це інтеграційні тести, які лежать між невеликими тестами та великими тестами. Вони інтегрують кілька компонентів, і вони працюють на емуляторах або реальних пристроях.

Великі тести - це інтеграція та тести інтерфейсу, що виконуються, заповнюючи робочий процес інтерфейсу користувача. Вони гарантують, що основні кінцеві користувачі працюють так, як очікувалося, від емуляторів або реальних пристроїв. Таке тестування можна виконувати вручну або засобами Espresso Testing Framework.

Окрім функціонального та юзабіліті тестування, веб-застосунки тестуються загальними тест-кейсами застосувань - зміна розміру екрану, згортання/розгортання застосунку, роботу у фоновому режимі, зміна браузера, зникнення та поява підключення до мережі, надання/ненадання користувачем дозволів, встановлення хибних параметрів, оновлення,

сповіщення, низький рівень здатності процесора, видалення частини застосунку тощо.

Також застосунок необхідно перевіряти на всіх тестах на пристроях різних моделей, різної версії ОС, різних розмірів екрану.

3.3 Опис тестів

Тестування, описане нижче, здійснюється для веб-застосунку системи обліку сировини та персоналу металооброблювального підприємства створеного з використанням мови JavaScript та бібліотеки Node.js. Пристрій - ноутбук або ПК має задовольняти наступні вимоги для стабільного користування застосунком:

Системні вимоги:

- Windows 7 Home або вище;
- сучасний браузер Chrome або Safari;

Апаратні вимоги:

- Процессор частотою від 1.8 ГГц;
- RAM від 4 ГБ;
- SSD або HDD 128 ГБ або більше
- Наявність достатнього вільного місця у пам’яті для встановлення.

Тестування здійснювалося на персональному комп’ютері з операційною системою Windows 7 Home та екраном Phoilips FullHD та ноутбук MacBook 13 (MacOS Siera).

Тестування встановлення описане у таблицях 3.1-3.2. Виконується на персональному комп’ютері з операційною системою Windows 7 Home та сучасним браузером Google Chrome 62.3.52.

Таблиця 3.1 – Встановлення застосунку

Мета тесту	Перевірка можливості встановлення застосунку.
Початковий стан	В браузер завантажено веб-застосунок
Вхідні дані	-

Продовження таблиці 3.1

Схема проведення тесту	Ввести в поле логін master@admim.com та 123qwe в поле паролю натиснути кнопку “Login”,
Очікуваний результат	Вхід в застосунок успішний.
Фактичний результат	Вхід в застосунок успішний

Таблиця 3.2 – Видалення застосунку та створених ним файлів

Мета тесту	Перевірка можливості видалення працівника та створених ним файлів.
Початковий стан	Застосунок встановлений.
Вхідні дані	-
Схема проведення тесту	Спробувати видалити працівника звичайним шляхов. Натиснути “Видалити”.
Очікуваний результат	Застосунок успішно видалено разом з файлами, що були створені ним під час роботи.
Фактичний результат	Застосунок успішно видалено разом з файлами, що були створені ним під час роботи.

Тестування використання описане в таблицях 3.3-3.14. Виконується на ноутбуці MacBook 13 (MacOS Siera) та персональному комп’ютері з операційною сиситемою Windows 7 Home та браузером Chrome 62.3.52.

Таблиця 3.3 – Перше використання

Мета тесту	Перевірка поведінки застосунку при першому використанні.
Початковий стан	Застосунок запущено. Сторінка “Login” відображається користувачу.
Вхідні дані	-
Схема проведення тесту	Відкрити застосунок.

Продовження таблиці 3.3

Очікуваний результат	Відкривається екран “Start screen”. На екрані кнопка “Login”. Так як користувач ще не розпочинав кешування текстового вмісту зображень, застосунок все ще вважається не використаним.
Фактичний результат	Відкривається екран “Start screen”. На екрані кнопка “Login”. Так як користувач ще не розпочинав кешування текстового вмісту зображень, застосунок все ще вважається не використаним.

Таблиця 3.4 – Запуск процесу першого додання сировини при умові що у системі немає наявних сировини.

Мета тесту	Перевірка поведінки застосунку при запуску процесу першого додання сировини.
Початковий стан	Відкритий екран “Login screen” після попереднього тесту.
Вхідні дані	-
Схема проведення тесту	Натиснути кнопку “Login”.
Очікуваний результат	Відкривається екран “Spinner”. На екрані присутні текст “Write your name” та Please wait, що показує прогрес виконання розпізнавання та кешування. Після завершення відкриється екран “Statistic”. На екрані присутні поле для вводу тексту з кнопкою “Search”, що стає доступною після вводу хоча б 3 букв.

Продовження таблиці 3.4

Фактичний результат	Відкривається екран “Spinner”. На екрані присутні текст “Please wait of loading” та progress bar, що показує прогрес виконання розпізнавання та кешування. Після завершення відкриється екран “Login”. На екрані присутні поле для вводу тексту з кнопкою “Search”, що стає доступною після вводу хоча б 3 букв.

Таблиця 3.5 – Запуск процесу першого розпізнавання зображень при умові що у пам’яті пристрою відсутні зображення

Мета тесту	Перевірка поведінки застосунку при запуску процесу першого розпізнавання зображень.
Початковий стан	Відкритий екран “Login screen” після попереднього тесту.
Вхідні дані	-
Схема проведення тесту	Натиснути кнопку “Login”.
Очікуваний результат	Відкривається екран “No images”. На екрані присутній текст “There is no images in your gallery. Add something to work with application”.
Фактичний результат	Відкривається екран “No images”. На екрані присутній текст “There is no images in your gallery. Add something to work with application”.

Таблиця 3.6 – Пошук зображень у застосунку

Мета тесту	Перевірка поведінки застосунку при запуску пошуку зображень у застосунку.
Початковий стан	Відкритий екран “Search” після попереднього тесту.

Продовження таблиці 3.6

Початковий стан	Відкритий екран “Search” після попереднього тесту.
Вхідні дані	Текст для пошуку.
Схема проведення тесту	Ввести текст у поле вводу. Натиснути “→”.
Очікуваний результат	Відкривається екран “Process”. На екрані присутні текст “Process might take a while” та progress bar, що показує прогрес пошуку. Після завершення відкривається екран “Results”. На екрані присутні текст “Search results”, де n - число знайдених результатів (0 якщо не знайдено), та ескізи знайдених зображень у 2 колонки.
Фактичний результат	Відкривається екран “Process”. На екрані присутні текст “Process might take a while” та progress bar, що показує прогрес пошуку. Після завершення відкривається екран “Results”. На екрані присутні текст “Search results”, де n - число знайдених результатів (0 якщо не знайдено), та ескізи знайдених зображень у 2 колонки.

Таблиця 3.7 – Відкриття зображення для редагування тексту або позначення як “Обране”

Мета тесту	Перевірка поведінки застосунку для відкриття статистики для редагування працівників або сировини
Початковий стан	Відкритий екран “Statistic” після тесту 2.4
Вхідні дані	Вибрана сировина.

Продовження таблиці 3.7

Схема проведення тесту	Натиснути на сировину довгим натисканням
Очікуваний результат	Відкривається екран “Image”, що містить зображення,розпізнанийтекст,кнопку редагування та кнопку позначити як “Нове”.
Фактичний результат	Відкривається екран “Image”, що містить зображення,розпізнанийтекст,кнопку редагування та кнопку позначити як “Нове”.

Таблиця 3.8 – Позначення зображення як “Обране”

Мета тесту	Перевірка поведінки застосунку
Початковий стан	Відкритий екран “Chat” після попереднього тесту.
Вхідні дані	-
Схема проведення тесту	Ввести текст та натиснути “Send”
Очікуваний результат	Повідомлення відправлено.
Фактичний результат	Повідомлення відправлено.

Таблиця 3.9 – Повторний вхід у застосунок

Мета тесту	Перевірка поведінки застосунку при повторному вході.
Початковий стан	Користувач вже користувався застосунком. З того часу у пам’яті пристрою з’явилися нові зображення.
Вхідні дані	-
Схема проведення тесту	Відкрити застосунок.

Продовження таблиці 3.9

Очікуваний результат	Відкриється екран “Upload”. На екрані присутні текст “Hello again! It looks like new images have appeared in your gallery.”
Фактичний результат	Відкривається екран “Upload”. На екрані присутні текст “Hello again! It looks like new images have appeared in your gallery. Update?” та 2 кнопки: “Yes” та “Skip this time”.

Таблиця 3.10 – Оновлення бази зображень у веб-застосунку

Мета тесту	Перевірка поведінки застосунку при оновленні бази сировини
Початковий стан	Відкритий екран “Update” після попереднього тесту.
Вхідні дані	-
Схема проведення тесту	Натиснути кнопку “Yes”.
Очікуваний результат	Відкривається екран “Process”. На екрані присутні текст “Process might take a while” та progress bar, що показує прогрес оновлення. Після завершення відкриється екран “Search”. Зміст нових зображень збережений у базі застосунку.
Фактичний результат	Відкривається екран “Process”. На екрані присутні текст “Process might take a while” та progress bar, що показує прогрес оновлення. Після завершення відкриється екран “Search”. Зміст нових зображень збережений у базі застосунку.

Продовження таблиці 3.11

Очікуваний результат	Відкритий екран “Search”. Зміст нових зображень не збережений у базі застосунку. При наступному вході у застосунок користувача буде знову запропоновано здійснити оновлення (екран “Update”).
Фактичний результат	Відкритий екран “Search”. Зміст нових зображень не збережений у базі застосунку. При наступномувходіузастосунокзнову запропоновано здійснити оновлення (екран “Update”).

Таблиця 3.13 – Розпізнавання зображення з камери в режимі реального часу

Мета тесту	Перевірка роботи розпізнавання зображення з камери в режимі реального часу
Початковий стан	Відкритий екран “Search”.
Вхідні дані	-
Схема проведення тесту	Натиснути кнопку “Realtime”.
Очікуваний результат	Відкритий екран “Realtimerecognition”. Зображення з камери розпізнається в режимі реального часу. Розпізнаний текст відображається внизу зображення та може бути скопійованийубуферобмінудовгим натисненням.
Фактичний результат	Відкритий екран “Realtimerecognition”. Зображення з камери розпізнається в режимі реального часу. Розпізнаний текст відображається внизу зображення та може бути скопійований.

Таблиця 3.14 – Оновлення бази зображень у застосунку, за умови що недостатньо вільного місця у пам’яті

Мета тесту	Перевірка поведінки застосунку при оновленні бази сировини, за умови що недостатньо вільного місця у пам’яті.
Початковий стан	Відкритий екран “Materials”, розпочалося оновлення бази даних.
Вхідні дані	-
Схема проведення тесту	Недостатньо вільного місця у пам’яті.
Очікуваний результат	Оновлення бази даних припиняється, на екрані з’являється повідомлення “Sorry. Your disk all full. Free some space to be able to update search database”.
Фактичний результат	Оновлення бази даних припиняється, на екрані з’являється повідомлення “Sorry. Your disk all full. Free some space to be able to update search database”.

Тестування сумісності. Виконання всіх попередніх Тестування встановлення та тестування зручності використання тестів на пристроях:

- MacBook 15 та ПК ;
- з операційною системою MacOS Siera та Windows 7 Home;
- з розмірами екрану 1920x1280 та 1280x600.

Результати тестів не відрізняються від очікуваних результатів, описаних у Тестування зручності використання.

Тестування виконання. Виконання всіх попередніх Тестування зручності використання тестів на MacBook 13 з низьким рівнем щільності точок.

Низький рівень заряду акумулятора пристрою не має впливати на правильність роботи застосунку. Результати тестів при низькому рівні заряду

акумулятора не відрізняються від очікуваних результатів, описаних у тестування зручності використання.

Тестування переривання. Виконання всіх попередніх Тестування зручності використання тестів на ПК з операційною системою Windows 7 Home з:

- перериванням згортанням та розгортанням застосунку;
- блокування/розблокування пристрою;
- зміна екрану;
- перериванням вхідним викликом або іншим застосунком.

Будь-які переривання не мають впливати на правильність роботи застосунку. Результати тестів з додавання переривань не відрізняються від очікуваних результатів, описаних у тестування зручності використання.

3.4 Висновки до розділу

Тестування веб-застосунків має свої особливості: різні розміри екрану, апаратні відмінності, кілька версій операційних систем, різні типи підключення до інтернету, раптові обриви інтернет зв'язку тощо. Тестування застосунку у процесі розробки надає значні переваги. До них можна віднести наступне: швидке виявлення помилок і несправностей, рефакторинг коду, стабільна швидкість розробки, що веде до зменшення витрат. Тільки послідовно виконуючи всі тести, можна підтвердити правильність, функціональну поведінку та зручність використання застосунку, перш ніж публікувати його.

У розділі наведені Тестування встановлення, Тестування зручності використання, Тестування сумісності, Тестування виконання та Тестування переривання тести для веб-застосунку пошуку зображень, збережених у пам’яті пристрою, по їх текстовому вмісту, створеного з використанням мови JavaScript та засобів розробки React.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для встановлення веб-застосунку потрібен ПК пристрій із встановленою операційною системою Windows 7 Home та Chrome 62.4.54. Потрібно завантажити на пристрій браузер програмне забезпечення для роботи з базою даних Compas та node.js бібліотеку для роботи з мовою програмування Javascript. Необхідно мати швидкісний доступ до інтернету, щоб завантажити всі необхідні модулі для коректної роботи з веб-застосунком. На рисунку 4.1 зображено команду для встановлення всіх залежностей.

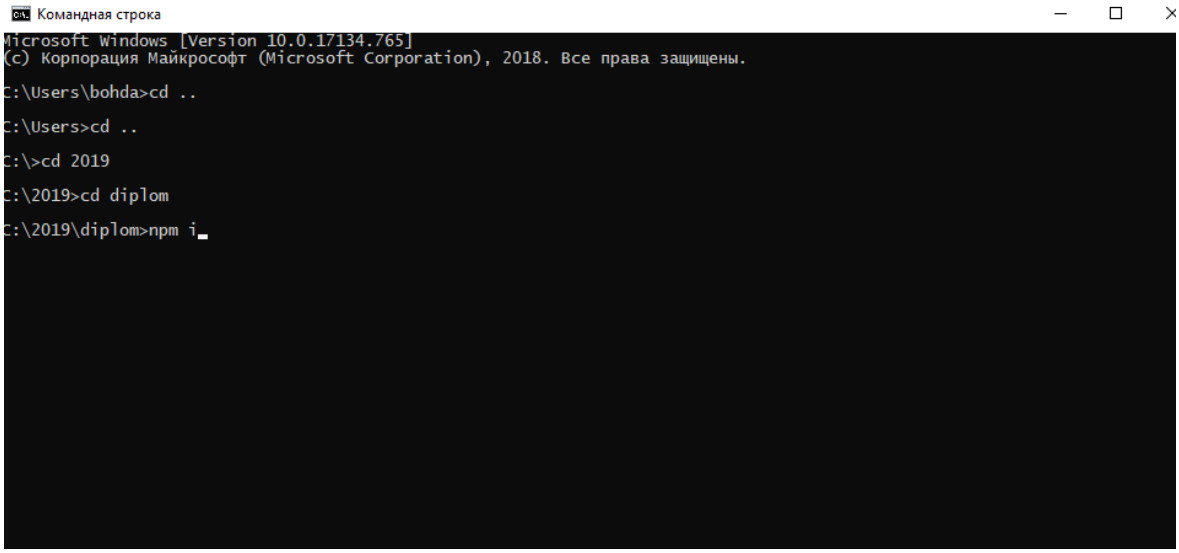
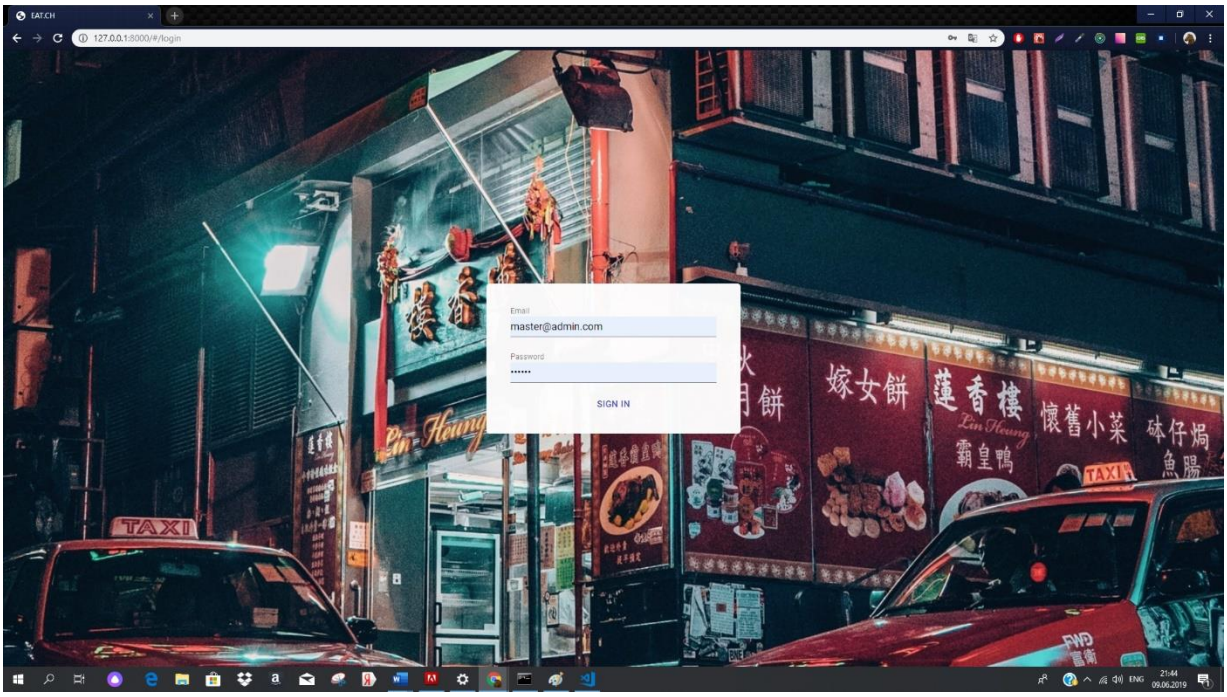


Рисунок 4.1 – Встановлення залежностей для React-застосунку

Після втановлення залежностей необхідно запустити проект за допомогою консолі. На рисунку 4.2 вказано комнду для запуску проекту. Після введення команди в браузері відкриється вкладка localhost:3000 де буде відображено систему обліку сировини та персоналу металооброблювального підприємства.

Рисунок 4.1 – Команда для запуску React-застосунку

Після компіляції та запуску проекта в браузері зявиться форма входу. сторінка входу в систему зображено на рисунку 4.3.



Для входу в систему необхідно ввести логін master@admin.com та пароль 123qwe. Після введення всіх полів на сторінці необхідно натиснути на кнопку “SIGN IN”. Першою сторінкою після входу в систему буде “Dashboard”, на цій сторінці зображено статистику та інформацію про стан підприємства. На рисунку 4.4 зображено як виглядає сторінка з статистикою.

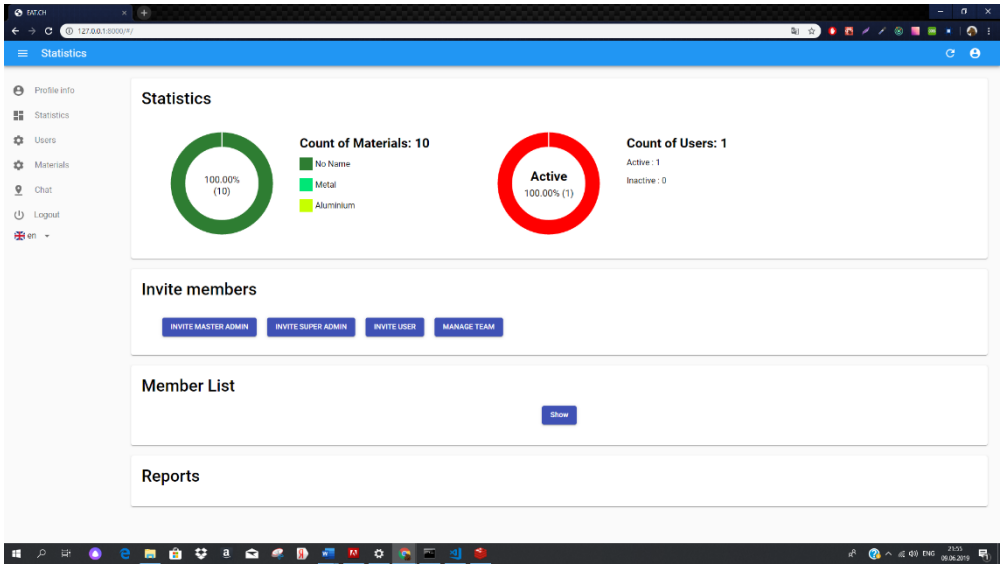


Рисунок 4.4 – Сторінка статистики

Після всіх вище вказаних дій необхідно перейти на сторінку “Profile” та заповнити всі поля.

4.2 Робота з програмним забезпеченням

Для запрошення нових користувачів в систему необхідно вибрати іхню роль в секції “Invite members” та натиснути на відповідну кнопку. Після чого заповнити всі необхіні поля та натиснути на кнопку “Invite”. Приклад запрошення для працівника зображено на рисунку 4.5.

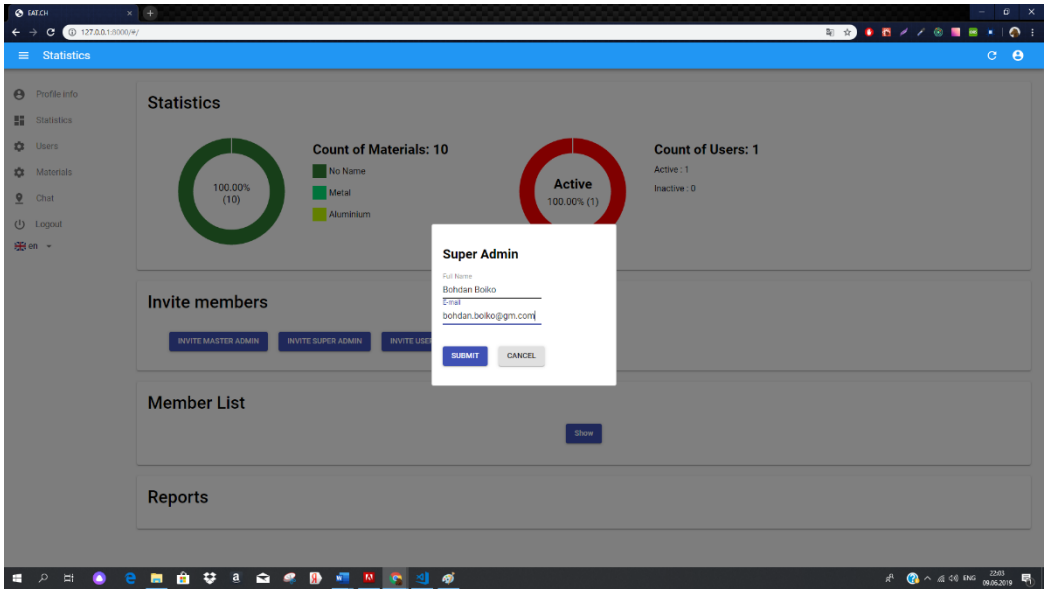


Рисунок 4.4 – Сторінка запрошення працівника

Перейшовши на сторінку “Materials” добавимо новий матеріал на склад, а потім видалемо один з списку. Ці дії зображено на рисунку 4.6.

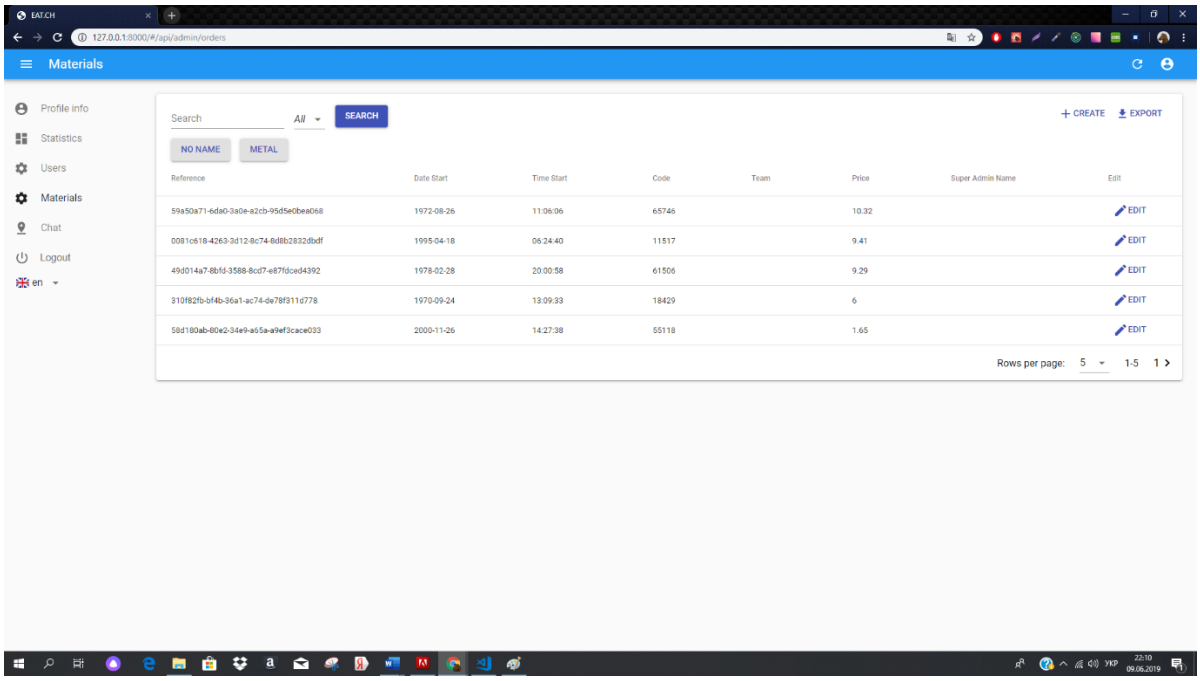


Рисунок 4.4 – Сторінка додавання нового матеріалу

4.3 Супровід програмного забезпечення

Після публікації у GitHub для контролю виникнення проблем, які не було відтворено на етапах розробки і тестування, використовуються засоби сервісу Google Console. Сервіс відображає статистику користування застосунком і логує помилки.

4.4 Висновки до розділу

У розділі описане розгортання застосунку, наведено інформацію стосовно користування веб-застосунком і вказані засоби супроводу веб-застосунку.

ВИСНОВКИ

У ході дипломного проекту було створено систему обліку сировини та персоналу металооброблювального підприємства. Були проаналізовані наявні технічні рішення та веб-застосунки, що працюють з обліком на підприємстві. Під час аналізу були виявлені їхні переваги та недоліки. Сформовано вимоги для застосунку.

Спроектований і розроблений веб-застосунок для пристроїв з будь-якою операційною системою та будь-яким сучасним браузером для полегшення роботи з збередення та аналізу інформації підприємства.

Він має ряд переваг перед іншими існуючими аналогами:

– не потрібно створювати окрему систему для підприємства, застосунок працює одразу усіма сферами;

– застосунок підтримує актуальний перелік сировини та працівників у пам’яті та автоматично пропонує користувачу оновити його;

– застосунок надає користувачу можливість пошуку, редагування та видалення працівників та сировини у базі даних, що значно полегшує роботу з їх обліком

– застосунок надає користувачу статистику підприємства в реальному часі.

Після завершення основного етапу розробки було проведено тестуванння та усунено знайдені недоліки.

Було розроблено проектну документацію, схеми варіантів використань, схеми функціональних вимог, схеми бізнес-процесів застосунку, схеми класів програмного забезпечення, програму та методику тестування, а також керівництво користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1) Створеня інтелектуальної системи обліку [Електронний ресурс] Режим доступу: http://elar.nung.edu.ua/bitstream/123456789/4980/1/3020s.pdf

2) React [Елек тронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/React_(JavaScript_library)

3) Angular [Електронний ресурс] – Режим доступу: https://angular.io/

4) Vue.js [Електронний ресурс] – Режим доступу: https://vuejs.org/

5) SAP ERP [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/SAP_ERP

6) Microsoft Dynamics [Електронний ресурс] – Режим доступу: https://dynamics.microsoft.com/en-us/

7) Best OCR Apps for Extracting Text [Електронний ресурс] – Режим доступу: https://www.makeuseof.com/tag/best -ocr-apps-extracting-text-images/

8) Evernote Tech Blog [Електронний ресурс] – Режим доступу: http://blog.evernote.com/tech/2013/07/18/how-evernotes-image-recognition-works/

9) StatCounter GlobalStats [Електронний ресурс] – Режим доступу: http://blog.evernote.com/tech/2013/07/18/how-e

10) React Developers Documentation and Guides [Електронний ресурс] – Режим доступу: https://developer.raect.com

11) Save data using SQLite [Електронний ресурс] – Режим доступу: https://developer.android.com/training/data-storage/sqlite

12) Enzyme Testing [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/mobile_testing/index.html

ДОДАТОК А

А.1 Програмний код головної сторінки

```
import React from 'react'
import './index.css'
import  { Provider } from 'react-redux'
import { BrowserRouter as Router, Route, Link } from 'react-router-dom'
import createAdminStore from './store/createAdminStore'
import createHistory from 'history/createHashHistory'
import customRoutes from './routers/cutomRoutes'
import { Admin, Resource } from 'react-admin'
import MyLoginPage from './pages/login'
import Dashboard from './pages/dashboard'
import LogoutButton from './components/LogoutButton'
import { DeliveryEdit, DeliveryCreate, DeliveryList } from './pages/delivery'
import MyLayout from './components/Layout'
import { UserList, UserEditComponent } from './pages/users'
import { OrdersList, OrderEdit } from './pages/orders'
import authProvider from './components/providers/authProvider'
import dataProvider from './components/DataProvider/dataProvider'
require('dotenv').config()

const history = createHistory()
const i18nProvider = locale => {
 if (locale !== 'en') {
  return []
 }
 return []
}
class App extends React.Component {
 render () {
  return (
   <Provider
    store={createAdminStore({
     authProvider,
     dataProvider,
     i18nProvider,
     history
    })}
   >
    <Router>
     <Admin
      title='Materials'
      dashboard={Dashboard}
      loginPage={MyLoginPage}
      logoutButton={LogoutButton}
      authProvider={authProvider}
       dataProvider={dataProvider}
      appLayout={MyLayout}
```

```
        customRoutes={customRoutes}
        >
          <Resource
            name='api/admin/drivers'
            options={{ label: 'Delivery Drivers' }}
            list={DeliveryList}
            create={DeliveryCreate}
            edit={DeliveryEdit}
          />
          <Resource
            name='api/admin/orders'
            options={{ label: 'Orders' }}
            list={OrdersList}
            edit={OrderEdit}
          />
          <Resource
            name='api/admin/users'
            options={{ label: 'Users' }}
            list={UserList}
            edit={UserEditComponent}
          />
        </Admin>
      </Router>
    </Provider>
  )
 }
}

export default App
```

А.2 Програмний код таблиці

```
import React from 'react'
import 'typeface-roboto'
import {connect} from 'react-redux'
import * as R from 'ramda'
import {getResources, showNotification} from 'react-admin'
import Table from '@material-ui/core/Table'
import Paper from '@material-ui/core/Paper'
import {DeletePopup, Footer, Header} from '../TableComponent'
import TableToolbar from '../TableComponent/Toolbar'
import dataProvider from '../DataProvider/dataProvider'
import TableBody from '@material-ui/core/TableBody'
import TableCell from '@material-ui/core/TableCell'
import TableRow from '@material-ui/core/TableRow'
import Checkbox from '@material-ui/core/Checkbox'
import Button from '@material-ui/core/Button'
import EditIcon from '@material-ui/icons/Edit'
import DeleteIcon from '@material-ui/icons/Delete'
import ChatIcon from '@material-ui/icons/Chat'
```

```
import ToggleButtonCustom from '../action/ToggleButtonCustom'
import OrderStatus from '../action/OrderStatus'
import SelectRole from '../action/SelectRole'
import {redTheme} from '../Layout/Themes'
import WalletInput from '../../components/ManageWalletInput'
import './style.css'
class CusomTable extends React.Component {
  state = {
    order: 'desc',
    field: 'id',
    selected: [],
    data: [],
    page: 1,
    perPage: 5,
    total: 5,
    filter_q: '',
    filter_name: '',
    delete_popup: false,
    delete_item: 0,
    order_status: [],
    walletCalculation: '',
    date: '',
    city: ''
  }
  generateParams = () => {
    let page = this.state.page
    let perPage = this.state.perPage
    let field = this.state.field
    let order = this.state.order
    let q = this.state.filter_q
    let city = this.state.city
    let date = this.state.date
    let order_status = this.state.order_status
    let name = this.state.filter_name === 'all' ? '' : this.state.filter_name
    return {
      pagination: {
        page,
        perPage
      },
      sort: {
        field,
        order
      },
      filter: {
        q,
        name
      },
      order_status,
      city,
      date
```

```
render () {
     const {
       onSelectAllClick,
       order,
     } = this.props
 return (
       <TableHead>
         <TableRow>
           {this.props.bulk && (
             <TableCell padding='checkbox'>
               <Checkbox
                 indeterminate={numSelected > 0 && numSelected < rowCount}
                 checked={numSelected === rowCount}
                 onChange={onSelectAllClick}
               />
             </TableCell>
           )}
           {this.props.rows.map(
             (row, index) => (
               <TableCell
                 key={index}
                 align={row.numeric ? 'right' : 'left'}
                 padding={row.disablePadding ? 'none' : 'default'}
                 sortDirection={orderBy === row.name ? order : false}
               >
                 <Tooltip
                   title='Sort'
                   placement={row.numeric ? 'bottom-end' : 'bottom-start'}
                   enterDelay={300}
                 >
                   <TableSortLabel
                     active={orderBy === row.name}
                     direction={order}
                     onClick={
                       row.filter ? this.createSortHandler(row.name) : () => {}
                     }
                   >
                     {row.label}
                   </TableSortLabel>
                 </Tooltip>
               </TableCell>
             ),
             this
           )}
         </TableRow>
       </TableHead>
     )
   }
 }
```

А.3 Програмний код статистики сировини

```
import React, { Component } from 'react'
import OrderStatus from '../../../../components/action/OrderStatus'
import './style.css'
let DoughnutChart = require('react-chartjs').Doughnut

class Circle extends Component {
  state = {
    status: '',
    value: 0,
    total: 0,
    activeColor: null
  }

  componentDidMount () {
    let sum = 0
    for (let key in this.props.data) {
      sum += this.props.data[key].value
    }
    this.setState({
      status: this.props.data[0].title,
      value: this.props.data[0].value,
      total: sum
    })
  }

  onClickFunc = evt => {
    let activePoints = this.refs.chart.getSegmentsAtEvent(evt)
    this.props.data[0].color = '#FF5A5E'
    if (activePoints[0] !== undefined) {
      this.setState(() => ({
        status: activePoints[0].title,
        value: activePoints[0].value
      }))
    }
  }

  render () {
    let persentage = ((this.state.value / this.state.total) * 100).toFixed(2)
    const {lang} = this.props
    return (
      <div className='statContainer'>
        <div className='statHeader'>
          <span className='statTitle' />
        </div>
        <div className='ordersChartBox'>
          <div className='dataBox'>
            <div className='coBox'>
              <DoughnutChart
```

```
                className='dashboard-chart'
                ref='chart'
                data={this.props.data}
                options={{ percentageInnerCutout: 70 }}
                width='200'
                height='200'
                onClick={this.onClickFunc}
              />
            </div>
            <div className='chartData'>
              <p className='circleHeader'>{lang[this.state.status]}</p>
              <p className='circleNumbers'>
                {persentage}% ({this.state.value})
              </p>
            </div>
          </div>
          <div className='ordersTextBox'>
            <p className='ordersHeader'>
              {this.props.label}: {this.state.total}
            </p>
            {this.props.orders
              ? this.props.data.map((item, index) => {
                return <OrderStatus lang={lang} key={index} order={item} />
              })
              : this.props.data.map((item, index) => {
                return (
                  <p key={index} className='ordersNumbers'>
                    {item.title} : {item.value}
                  </p>
                )
              })}
          </div>
        </div>
      </div>
    )
  }
}

export default Circle
```

А.4 Програмний код чату

```
import React, {Component, Fragment} from 'react'
import './chat.css'
import PropTypes from 'prop-types'
import Echo from 'laravel-echo'
import {compose} from 'recompose'
import {connect} from 'react-redux'
import {withStyles} from '@material-ui/core/styles'
import ChatUserList from './UserList'
```

```
import {
    addMessage,
    clearMessages,
    fetchChatUsers,
    fetchMessages,
    filterChatUsers
} from '../../store/actions/chat'
import {createRoom, getRoomList} from '../../api/chat'
import * as R from 'ramda'

const styles = theme => ({
    chatMessagesWrapper: {
        display: 'flex',
        justifyContent: 'center',
        alignItems: 'center',
        flexBasis: '70%'
    }
})

class Chat extends Component {
    state = {
        value: '',
        isOpenChat: false,
        selectedUser: {},
        rooms: [],
        selectedChat: ''
    }

    componentDidMount () {
        const token = localStorage.getItem('token')
        const authUserId = localStorage.getItem('id')
        this.props.fetchChatUsers()
        this.getRooms()
        window.io = require('socket.io-client')
        window.Echo = new Echo({
            broadcaster: 'socket.io',
            host: window.location.hostname + ':6001',
            auth: {
                headers: {Authorization: 'Bearer ' + token}
            }
        })
        console.log(12)
        window.Echo.private(`User.${authUserId}`).listen('.room.created', event
=> {
            console.log('eeeeee2222vent', event)
            this.props.clearMessages([event['last_message']])
            this.getRooms()
        })
    }
```

```
componentDidUpdate (prevProps, prevState, snapshot) {
    const {rooms} = this.state
    const authUserId = localStorage.getItem('id')
    if (R.equals(prevState.rooms, rooms)) {
        return null
    } else {
        console.log('refreshed')
        console.log(window)

        rooms.forEach(room => {
            window.Echo.join(`Chat.${room.id}`)
                .here(users => {
                    console.log(users)
                })
                .joining(user => {
                    console.log(user.name + 'j')
                })
                .leaving(user => {
                    console.log(user.name + 'l')
                })
                .listen('.message.sent', e => {
                    console.log('123123123123123', e)
                    !R.equals(Number(authUserId), e['sender_id']) &&
                    this.props.addMessage(e)
                })
        })
    }
}

componentWillUnmount () {
    this.props.clearMessages()
}

/**
 * Open Chat handler. Create room with selected drivers
 * @param id
 */
handleShowChat = id => {
    const {users} = this.props
    const {rooms} = this.state

    /**
     * Find user by id helper
     * @type {f1}
     */
    const selectedUser = R.find(R.propEq('id', id), users)

    /**
     * Check the presence of the user in the chat-room helper
     * @param room
```

```
 const roomId = !R.isNil(driverRoom) && driverRoom['id']
    if (R.includes(driverRoom, rooms)) {
        this.props.fetchMessages(roomId)
    } else {
        createRoom({
            reciever_id: id
        })
        this.props.clearMessages()
    }
    this.setState({
        isOpenChat: true,
        selectedUser: selectedUser
    })
}

/**
 * Fetch all chat-roms with drivers
 */
getRooms = () => {
    getRoomList()
        .then(res => {
            this.setState({
                rooms: res.data
            })
        })
        .catch(err => console.log(err.response))
}

handleCloseChat = () => {
    this.setState({
        isOpenChat: false
    })
    this.props.clearMessages()
}

/**
 * Filtering users by input field value
 * @param event
 */
handleFilterUsers = event => {
    this.props.filterChatUsers(event.target.value)
}

render () {
    const {users} = this.props
    const {isOpenChat, selectedUser, rooms, lang, role} = this.state
    return (
        <div className='chat__wrapper'>
            {
                !R.equals(role, 'super_admin') ? <Fragment>
```

```
                                handleCloseChat={this.handleCloseChat}
                                isOpenChat={isOpenChat}
                                rooms={rooms}
                                selectedUser={selectedUser}
                            />
                        </Fragment>
                        : <div>Permission denied</div>
                }
            </div>
        )
    }
}

Chat.propTypes = {
    users: PropTypes.array.isRequired
}

const mapStateToProps = state => {
    return {
        users: getAllUsers(state),
        lang: state.i18n.messages,
        role: state.login.role
    }
}

export default compose(
    connect(
        mapStateToProps,
        {
            fetchChatUsers,
            filterChatUsers,
            fetchMessages,
            addMessage,
            clearMessages
        }
    ),
    withStyles(styles)
)(Chat)
```

А.5 Програмний код сторінки обліку сировини

```
import React from 'react'
import Card from '@material-ui/core/Card'
import {Title} from 'react-admin'
import TextField from '@material-ui/core/TextField'
import dataProvider from '../../components/DataProvider/dataProvider'
import MapDrawRoute from '../../components/action/MapDrawRoute'
import {connect} from 'react-redux'
import {showNotification} from 'react-admin'
import Button from '@material-ui/core/Button'
```

```
import Button from '@material-ui/core/Button'
import AddressWithPoints from
'../../components/AutoSaveFields/addressWithPoints'
import './style.css'
import * as R from 'ramda'
import {setActiveAddressField, setSearcDropoffhAddress, setSearcPickuphAddress}
from '../../store/actions/orders'

require('dotenv').config()

class OrderEdit extends React.Component {
    state = {
        data: {},
        pickupAddress: '',
        dropOffAddress: '',
        address: '',
        active: ''
    }

    componentDidMount () {
        dataProvider('GET_ONE', `${this.props.resource}`, {
            id: this.props.id
        })
            .then(res => {
                this.setState({
                    data: res.data
                })

this.props.setSearcPickuphAddress(res.data['pickup'].location['streetaddress'])

this.props.setSearcDropoffhAddress(res.data['dropoff'].location['streetaddress']
)
            })
            .catch(err => console.log(err))
    }

    handleChange = address => {
        R.equals(this.state.active, 'pickup') ?
            this.setState({
                pickupAddress: address
            }) : this.setState({
                dropOffAddress: address
            })
    }

    handleSetActive = name => {
        !R.isEmpty(name) ? this.setState({
            active: name
        }) : this.setState({
            active: ''
```

```
render () {
        const {data, active} = this.state
        const {lang} = this.props
        return (
            <Card className='driver__cards'>
                <Title title={lang.ordersEditTitle}/>
                {
                    !R.isEmpty(data) &&
                    <div className='simple-autosubmit__list'>
                        <div className={'topLine'}>
                            <h2>{lang.ordersInfoTitle}</h2>
                        </div>
                        <div className='simple-autosubmit__row'>
                            <div className={'textField'}>
                                <TextField
                                    disabled
                                    className={'field'}
                                    id='standard-disabled__small'
                                    label={lang.orderLabelDropOffTime}
                                    value={
                                        data['dropoff_time_from']
                                            ? data['dropoff_time_from']
                                            : ''
                                    }
                                    margin='normal'
                                />
                            </div>
                            <div className={'textField'}>
                                <TextField
                                    disabled
                                    className={'field'}
                                    id='standard-disabled'
                                    label={lang.tableReference}
                                    value={
                                        this.state.data['reference_id'] ?
this.state.data['reference_id'] : ''
                                    }
                                    margin='normal'
                                />
                            </div>
<div className={'textField'}>
                                <TextField
                                    disabled
                                    className={'field'}
                                    id='standard-disabled'
                                    label={lang.orderLabelPickupDate}
                                    margin='normal'
                                />
                            </div>
```